Cluster Rocks SOL

Manual de Usuario

Capítulo 1

Introducción a la plataforma

Índice

0.-Introducción

Con este manual se pretende introducir al usuario en la utilización de algunas de las herramientas que cuenta la plataforma, tanto para la ejecución de cálculo científico, como para tareas de monitorización y control.

El usuario debe estar familiarizado con el uso del sistema operativo Linux, aunque no se requieren conocimientos avanzados.

El manual está estructurado en una serie de capítulos en los que se abordan distintos aspectos relacionados con el uso del cluster. Ante todo se pretende darle una orientación netamente práctica, obviando aspectos que puedan entorpecer el manejo por parte del usuario.

1.- Acceso al frontend y a los nodos del cluster

Rocks es una colección de software de código abierto para crear un cluster sobre Linux. Todo cluster linux configurado con Rocks tiene dos tipos de máquinas: el frontend, donde se centraliza la información sobre la plataforma, se crean las cuentas de usuario y se ejecutan los servicios principales del cluster y los nodos de cómputo, que son las máquinas en las que se realizan los trabajos.

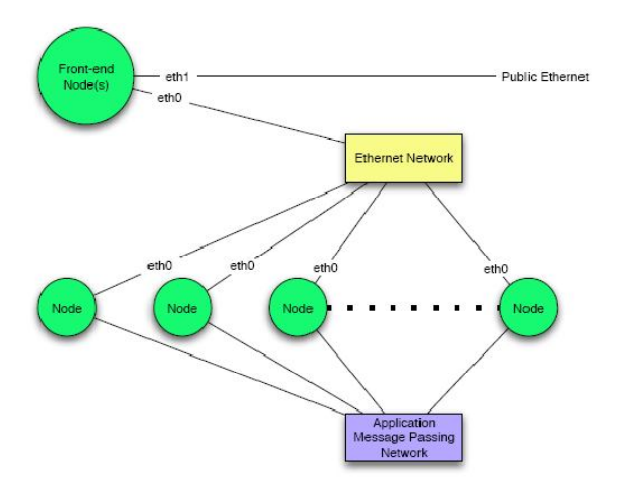


Figura 1: Arquitectura de un cluster Rocks

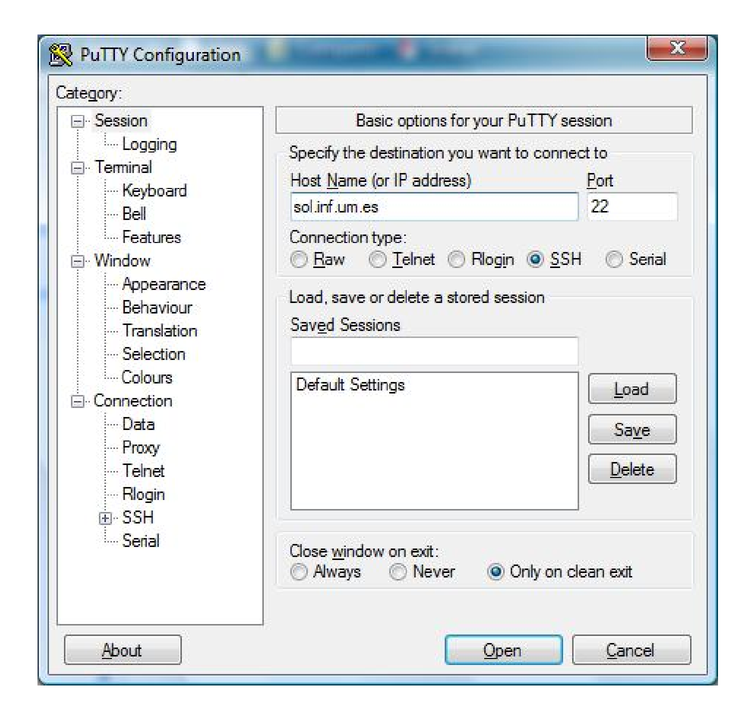
El acceso al frontend se hace solicitando una terminal remota a través del protocolo SSH. La mayoría de las máquinas Linux vienen con un cliente ssh que se invoca directamente desde el símbolo del sistema con el comando ssh.

```
$ ssh usuario@sol.inf.um.es
```

Desde Windows también es posible abrir una terminal remota. Uno de los clientes más populares es putty, que puede encontrarse para todas las versiones de Windows 95, 98, ME, NT, 2000, XP y Vista en la siguiente página:

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html.

La configuración del cliente es bastante sencilla, a continuación se muestra una captura del mismo.



Una vez conectados es recomendable cambiar inmediatamente la contraseña. Para cambiarla se utiliza el comando “passwd”.

```
$ passwd
```

Desde el frontend es posible acceder a cualquiera de los nodos de cómputo del cluster utilizando ssh. Rocks utiliza la convención de llamar a los nodos utilizando el prefijo “compute-x”, donde x es el número de cada nodo. Los nodos comienzan a numerarse en 0 y se encuentran inscritos en el dominio “.local”. Así, los nombres de los nodos de cómputo del cluster serán compute-0-0.local, compute-0-1, compute-0-2, compute-0-3-local,...

Rocks define unos sobrenombres para los nodos del cluster. Estos aliases se contruyen con el sufijo c0 y el número del cluster. Por ejemplo el nodo compute-0-0.local es equivalente c0-0.

La lista de todas los nodos del cluster con sus IPs se encuentra en el archivo /etc/hosts.

```
#
# Do NOT Edit (generated by dbreport)
#
127.0.0.1     localhost.localdomain     localhost
```

```
10.1.1.1      sol.local sol # originally frontend-0-0
10.255.255.254      compute-0-0.local compute-0-0 c0-0
10.255.255.253      compute-0-1.local compute-0-1 c0-1
10.255.255.252      compute-0-2.local compute-0-2 c0-2
10.255.255.251      compute-0-3.local compute-0-3 c0-3
155.54.204.142      sol.inf.um.es
```

Para acceder a uno de los nodos del cluster podemos elegir cualquiera de los siguientes comandos:

```
$ ssh compute-0-x.local
$ ssh c0-x
```

Donde x es el número del nodo.

Antes de empezar a trabajar es interesante verificar que los nodos están operativos. Esto se puede realizar usando el comando "ping".

```
$ ping -c 2 compute-0-x.local

[jgpicon@sol ~]$ ping -c 2 compute-0-2.local
PING compute-0-2.local (10.255.255.252) 56(84) bytes of data.
64 bytes from compute-0-2.local (10.255.255.252): icmp_seq=0 ttl=64 time=0.268
ms
64 bytes from compute-0-2.local (10.255.255.252): icmp_seq=1 ttl=64 time=0.229
ms

--- compute-0-2.local ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.229/0.248/0.268/0.025 ms, pipe 2
```

Este comando envía un paquete de internet y espera que le sea devuelto. Además calcula el timempo de ida y vuelta del paquete.

2.- Sistemas de archivos en el cluster

Los archivos que usa el usuario son almacenados automáticamente en su directorio casa. El directorio casa es normalmente /home/usuario. Rocks usa dos mecanismos básicos para garantizar que el usuario tenga acceso directo a sus archivos aún si se encuentra conectado a uno de los nodos de cómputo. De un lado está el NFS (Network filesystem) que monta el sistema de archivos de la cuenta del usuario a través de la red en el nodo al que se conecta(En el sistema NFS los cambios que se hacen sobre el sistema de archivos se actualizan automáticamente en el disco duro del frontend, donde residen realmente). El otro es el servicio autofs que garantiza que el montado de los sistemas de archivos sea automático y ocurra en el momento de acceso del usuario. Autofs también desmonta el sistema de archivos cuando el usuario deja de utilizarlo.

3.- El comando Cluster-fork

A menudo queremos ejecutar trabajos paralelos de comandos UNIX estandar. Por paralelos nos referimos al mismo comando ejecutandose en varios nodos del cluster. Esto podría ser útil para mover ficheros, ejecutar pequeños test y realizar varias tareas administrativas.

Rocks proporciona una comando muy útil para tales propósitos, el llamado cluster-fork. Por ejemplo, para hacer un listado de todos los procesos en los nodos del cluster escribiríamos lo siguiente:

```
$ cluster-fork “ps”
```

```
[jgpicon@sol ~]$ cluster-fork "ps"
compute-0-0:
  PID TTY          TIME CMD
28184 ?        00:00:00 sshd
28185 ?        00:00:00 ps
compute-0-1:
  PID TTY          TIME CMD
23385 ?        00:00:00 sshd
23386 ?        00:00:00 ps
compute-0-2:
  PID TTY          TIME CMD
22647 ?        00:00:00 sshd
22648 ?        00:00:00 ps
compute-0-3:
  PID TTY          TIME CMD
30589 ?        00:00:00 sshd
30590 ?        00:00:00 ps
```

Otro ejemplo muy útil para ver quien está conectado y qué está haciendo sería el siguiente:

```
$ cluster-fork “w”
```

```
[jgpicon@sol ~]$ cluster-fork "w"
cluster-fork "w"
compute-0-0:
 21:57:57 up 18 days,  9:20,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-1:
 21:57:57 up 19 days,  6:56,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-2:
 21:57:58 up 19 days,  9:24,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
compute-0-3:
 21:57:58 up 18 days,  9:23,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
```

Cuando se ejecuta un comando de la familia cluster, es posible seleccionar los nodos sobre los que deberá ejecutarse la acción o el comando seleccionado.

```
$ cluster-fork –n “c0-0 c0-3” w
```

En el ejemplo anterior la opción –n indica que, a continuación, entre comillas vienen especificados los nodos en los que se ejecutará la acción.

4.- Monitorización de recursos

Existen 3 tipos de recursos que pueden consultarse cuando se quiere usar el cluster: CPU, RAM y HD.

4.1.- CPU

El uso de la CPU es uno de los parámetros más importantes que pueden monitarizarse. Existen dos tipos de parametros: parámetros estáticos (nº de procesadores, velocidad de reloj,…) y parámetros dinámicos (carga del procesador).

Para conocer los parámetros estáticos podemos consultar el archivo /proc/cpuinfo

```
processor	: 0
vendor_id	: GenuineIntel
cpu family	: 15
model		: 6
model name	:                  Intel(R) Xeon(TM) CPU 3.00GHz
stepping	: 4
cpu MHz		: 2992.649
cache size	: 2048 KB
physical id	: 0
siblings	: 2
core id		: 0
cpu cores	: 2
fpu		: yes
fpu_exception: yes
cpuid level	: 6
wp		: yes
flags		: fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm syscall nx lm pni
monitor ds_cpl est cid cx16 xtpr
bogomips	: 5990.17
clflush size : 64
cache_alignment	: 128
address sizes: 36 bits physical, 48 bits virtual
power management:

processor	: 1
vendor_id	: GenuineIntel
cpu family	: 15
model		: 6
model name	:
```

…

En este archivo podemos consultar información como la velocidad de la CPU, nº de núcleos, memoria cache… (Notar que una máquina que use tecnología hyperthreading parecerá tener 2 procesadores efectivos, aunque realmente solo tenga uno con capacidades para procesamiento paralelo).

También es posible consultar estos mismos parámetros con el comando cluster-fork:

```
$ cluster-fork –n “c0-1 c0-2” “grep –A 8 processor /proc/cpuinfo”
```

El parámetro dinámico más importante se conoce como carga promedio del procesador (ladavg). La carga promedio del procesador se define como el número efectivo de procesos

que están en cola en un período de tiempo definido. Existen 3 loadavgs y cada uno mide la carga de los últimos 5, 10 y 15 minutos.

Una de de las formas de consultar este parámetro es mediante el comando “uptime”:

```
$ uptime

[jgpicon@sol ~]$ uptime

 22:27:29 up 19 days, 12:56,  1 user,  load average: 0.00, 0.00, 0.00
```

El primer número indica la hora de reloj de la máquina, el segundo el tiempo que la máquina lleva encendida, el tercer el número de usuarios que tienen una terminal abierta en ese momento y los tres últimos números indican la carga de los últimos 5, 10 y 15 minutos. Puede ser interesante consultar la carga promedio en todos los nodos del cluster, para ello escribiremos:

```
$ cluster-fork “uptime”

[jgpicon@sol ~]$ cluster-fork "uptime"

compute-0-0:

22:56:43 up 18 days, 10:19,  1 user,  load average: 0.00, 0.00, 0.00

compute-0-1:

22:56:44 up 19 days,  7:55,  0 users,  load average: 0.00, 0.00, 0.00

compute-0-2:

22:56:45 up 19 days, 10:22,  0 users,  load average: 0.00, 0.00, 0.00

compute-0-3:

22:56:45 up 18 days, 10:21,  0 users,  load average: 0.00, 0.00, 0.00
```

4.2.- RAM

En este caso también podemos determinar dos tipos de parámetros, estáticos (tamaño total de memoria RAM, tamaño de la memoria de swap) y dinámicos (% de RAM utilizado, % de memoria swap utilizada).

Los parámetros estáticos de la memoria RAM se pueden consultar directamente en el archivo /proc/meminfo

```
MemTotal:      1536812 kB
MemFree:          9152 kB
Buffers:         91396 kB
Cached:        1090444 kB
SwapCached:          0 kB
Active:         412364 kB
Inactive:      1037364 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:      1536812 kB
LowFree:          9152 kB
```

```
SwapTotal:     1020116 kB
SwapFree:      1019956 kB
Dirty:              12 kB
Writeback:           0 kB
Mapped:         300940 kB
Slab:            50316 kB
CommitLimit:   1788520 kB
Committed_AS:   800348 kB
PageTables:       7456 kB
VmallocTotal: 536870911 kB
VmallocUsed:    263484 kB
VmallocChunk: 536607103 kB
HugePages_Total:     0
HugePages_Free:      0
Hugepagesize:     2048 kB
```

Es posible consultar estos mismos parámetros de otros nodos mediante cluster-fork.

```
$ cluster-fork -n "c0-0 c0-1" "grep -A 1 MemTotal /proc/meminfo"
```

```
[jgpicon@sol ~]$ cluster-fork -n "c0-0 c0-1" "grep -A 1 MemTotal
/proc/meminfo"

c0-0:
MemTotal:      2053924 kB
MemFree:       1419164 kB
c0-1:
MemTotal:      1536816 kB
MemFree:        628496 kB
```

En el ejemplo anterior se ha consultado la memoria total de los nodos c0-0 y c0-1.

4.3.- HD

Finalmente podemos consultar los parámetros estáticos del disco (nº de particiones, directorio de montaje del sistema de archivos, tamaño de la patición,...) y los parámetros dinámicos, como el espacio disponible en la partición.

Para consultar el número de particiones, su punto de montaje y el tamaño total de esas particiones es posible usar el comando df.

```
$ df –h
```

```
[jgpicon@sol ~]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             7.7G  6.8G  597M  93% /
none                  751M     0  751M   0% /dev/shm
/dev/sda5             446G  4.6G  419G   2% /state/partition1
/dev/sda2             3.9G  464M  3.2G  13% /var
tmpfs                 363M  2.6M  361M   1% /var/lib/ganglia/rrds
/state/partition1/home/condor
                      446G  4.6G  419G   2% /home/condor
/state/partition1/home/javiercm
                      446G  4.6G  419G   2% /home/javiercm
/state/partition1/home/domingo
                      446G  4.6G  419G   2% /home/domingo
/state/partition1/home/jgpicon
                      446G  4.6G  419G   2% /home/jgpicon
```

La opción –h muestra los datos en un formato que se lee más fácilmente (k, GB).

Solo las particiones del disco /dev/sda son de interés. De ellas vale la pena resaltar la partición /state/partition1 del frontend que contiene los directorios home de los usuarios. El tamaño de esta partición define la capacidad total para almacenamiento de información de los usuarios.

De nuevo es posible consultar el tamaño total de las particiones de los nodos de cómputo del cluster usando cluster-fork.

```
$ cluster-fork -n "c0-0 c0-1" "df -h"
```

```
[jgpicon@sol ~]$ cluster-fork -n "c0-0 c0-1" "df -h"
c0-0:
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             7.7G  4.7G  2.7G  64% /
none                 1003M     0 1003M   0% /dev/shm
/dev/sda5              98G   92M   93G   1% /state/partition1
/dev/sda2             3.9G   96M  3.6G   3% /var
sol.local:/export/home/domingo
                      446G  4.6G  419G   2% /home/domingo
sol.local:/export/home/jgpicon
                      446G  4.6G  419G   2% /home/jgpicon
c0-1:
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             7.7G  4.7G  2.7G  64% /
none                  751M     0  751M   0% /dev/shm
/dev/sda5              98G   92M   93G   1% /state/partition1
/dev/sda2             3.9G   88M  3.6G   3% /var
sol.local:/export/home/domingo
                      446G  4.6G  419G   2% /home/domingo
sol.local:/export/home/jgpicon
                      446G  4.6G  419G   2% /home/jgpicon
```

5.- Monitorización de procesos

Una instrucción básica para la monitorización de procesos en el cluster la constituye el comando ps.

```
$ ps caux
```

Muestra todos los procesos en ejecución de una máquina. La lista de procesos devueltos por este comando puede ser muy grande y es necesario filtrar usando determinados criterios. Si, por ejemplo, se quisieran monitorizar los procesos de un usuario concreto podría usarse:

```
$ ps caux | grep usuario
```

```
[jgpicon@sol ~]$ ps caux | grep jgpicon

jgpicon  13580  0.0  0.1 38292 2000 ?        S    Oct29   0:00 sshd
jgpicon  13581  0.0  0.1 57208 1704 pts/5    Ss+  Oct29   0:00 bash
jgpicon  13786  0.0  0.0 50280  880 pts/5    T    Oct29   0:00 man
jgpicon  13789  0.0  0.0 53816 1092 pts/5    T    Oct29   0:00 sh
jgpicon  13790  0.0  0.0 53816  468 pts/5    T    Oct29   0:00 sh
jgpicon  13795  0.0  0.0 51464  676 pts/5    T    Oct29   0:00 less
jgpicon  14263  0.0  0.1 38148 1896 ?        S    Oct29   0:00 sshd
jgpicon  14264  0.0  0.0 55028 1528 pts/6    Ss   Oct29   0:00 bash
jgpicon  14446  0.0  0.1 18640 2188 pts/6    S+   Oct29   0:00 ssh
jgpicon  14451  0.0  0.1 37104 1796 ?        S    Oct29   0:00 sshd
jgpicon  14452  0.0  0.0 55028 1528 pts/7    Ss+  Oct29   0:00 bash
jgpicon  14970  0.0  0.1 38292 1904 ?        S    Oct29   0:00 sshd
```

```
jgpicon  14971  0.0  0.0 55028 1536 pts/8    Ss   Oct29   0:00 bash
```

La información que devuelve este comando es la siguiente: el nombre de usuario, el PID, el % de CPU utilizad, el % de la memoria utilizada, la cantidad de memoria virtual usada, la cantidad de memoria RAM utilizada, el estado del proceso, la fecha de inicio, la cantidad de tiempo que ha estado en ejecución y el comando abreviado asociado al proceso.

Puede que mucha de la información mostrada por la instrucción anterior no nos sea de utilidad y por tanto no queramos mostrarla. Para ello es posible indicarle qué información mostrar mediante el uso de la opción –o.

```
[jgpicon@sol ~]$ ps a -o pid,user,cmd | grep jgpicon
```

El ejemplo anterior nos muestra solamente el pid, el nombre de usuario y el nombre completo del comando de los procesos que ejecuta jgpicon.

Finalmente otro instrucción de interés es:

```
$ ps r –A
```

Que muestra los procesos en ejecución de una máquina. Para mostrar todos los procesos en ejecución en algunos nodos del cluster se usaría la instrucción cluster-fork.

```
$  cluster-fork –n “c0-0 c0-1 c0-2 c0-3” “ps r –A”
```

Esta instrucción podemos acompañarla con opciones como las vistas unas líneas atrás con el fin de filtrar la información que se muestra en pantalla. Por ejemplo, si solamente queremos mostrar el usuario y el tiempo que lleva ejecutándose el proceso escribiríamos:

```
$ cluster-fork -n "c0-0 c0-1" "ps r -A -o user,%cpu,etime"
```

```
cluster-fork -n "c0-0 c0-1" "ps r -A -o user,%cpu,etime"
c0-0:
USER     %CPU     ELAPSED
jgpicon   2.0       00:01
c0-1:
USER     %CPU     ELAPSED
jgpicon   2.0       00:01
…
```

6.- Resumen de comandos

Comandos	Descripción
$ ssh usuario@sol.inf.um.es	Conexión al cluster con el cliente ssh
$ passwd	Cambia la contraseña
$ ssh compute-0-x.local $ ssh c0-x	Conexión por ssh al nodo x del cluster
$ ping -c 2 compute-0-x.local	Verificación de la conectividad con un nodo del cluster
$ cluster-fork “ps”	Listado de procesos del cluster
$ cluster-fork “w”	Muestra información de quién está conectado.
$ cluster-fork –n “c0-0 c0-3” w	Igual que el anterior commando, pero restringido a los nodos indicados
$ cluster-fork –n “c0-1 c0-2” “grep –A 8 processor /proc/cpuinfo”	Determina las propiedades estáticas de la CPU de los nodos indicados
$ uptime	Determina las propiedades dinámicas de la CPU
$ cluster-fork “uptime”	Igual que el anterior comando, pero para todos los nodos
$ cluster-fork -n "c0-0 c0-1" "grep -A 1 MemTotal /proc/meminfo"	Consulta las propiedades estáticas de la RAM en el cluster
$ df –h	Determina las propiedades de los sistemas de archivos montados en linux.
$ cluster-fork -n "c0-0 c0-1" "df -h"	Determina las propiedades de los sistemas de archivos montados en el cluster.
$ ps caux	Muestra todos los procesos de la maquina
$ ps caux | grep usuario	Muestra todos los procesos de la máquina pertenecientes a usuario
$ ps r –A	Muestra todos los procesos en ejecución
$ cluster-fork –n “c0-0 c0-1 c0-2 c0-3” “ps r –A”	Muestra todos los procesos en ejecución en los nodos c0-0,...,c0-3
$ cluster-fork -n "c0-0 c0-1" "ps r -A -o user,%cpu,etime"	Muestra los procesos en ejecución en los nodos co-0 y co-1, mostrando el usuario, el % de CPU utilizado y el tiempo que lleva en ejecución